Corrigé des travaux dirigés d'Analyse Numérique - GME5 1^{e} année ingénieur ENSGSI

G. Vinsard

2010

1 Corrigé des travaux dirigés

1.1 No 1 et 2 : Calcul de points équidistants sur une courbe

L'objectif du TD est d'élaborer un algorithme de calcul de points equidistants sur une courbe 2D paramétrée a priori quelconque et de d'implanter cet algorithme en langage de Maxima.

Question 1 Compréhension du problème. Soit les courbes parametrées d'équation

$$\text{Demi cercle : } \begin{cases} x(t) = \cos t \\ \\ y(t) = \sin t \end{cases} \quad 0 < t < \pi \quad ; \text{ parabole : } \begin{cases} x(t) = t \\ \\ y(t) = t^2/2 \end{cases} \quad 0 < t < \pi$$

tracer qualitativement ces courbes et placer dessus les points de coordonnées

$$(x(t_0), y(t_0)), (x(t_1), y(t_1)), \ldots, (x(t_{N+1}), y(t_{N+1}))$$

où $t_0 = 0$, $t_{N+1} = \pi$ et, si on pose $d(t_n, t_p) = \sqrt{(x(t_n) - x(t_p))^2 + (y(t_n) - y(t_p))^2}$,

$$d(t_0, t_1) = d(t_1, t_2) = \ldots = d(t_N, t_{N+1}) \tag{1}$$

dans les cas $N = 1$ et $N = 2$.

Voir les dessins faits en séance.

Question 2 Dans le cas $N = 1$, écrire un algorithme de calcul permettant d'obtenir le point de paramètre t_1 tel que $d(t_0, t_1) = d(t_1, t_2)$ pour une courbe paramétrée quelconque et en interdisant toute différentiation des fonctions $x(t)$ et $y(t)$.

L'algorithme est mis sous forme de fonction

Algorithme 1 : Recherche de t_1 tel que $d(t_0, t_1) = d(t_1, t_2)$

TrouveLeMilieu$(x(t), y(t), t0, t2, \epsilon)$;
début
  $x_0, y_0 \longleftarrow x(t_0), y(t_0)$;
  $x_2, y_2 \longleftarrow x(t_2), y(t_2)$;
  $t_1 \longleftarrow$
  ResolutionParDichotomie$(x(t)-x_0)^2+(y(t)-y_0)^2-(x(t)-x_2)^2+(y(t)-y_2)^2, t_0, t_2, \epsilon)$;
  t_1 en sortie
fin

où « *ResolutionParDichotomie* est la fonction définie par l'algorithme 1 du cours.

Question 3 Dans le cas général, écrire l'algorithme de calcul permettant d'obtenir les N paramètres $t_1, \ldots, t_N$ de (1) en s'appuyant sur l'algorithme de la question précédente. Comme il est difficile d'utiliser directement N comme entrée du problème, on choisira plutôt la « *distance moyenne entre les points* » qui devra être définie.

Supposons les N points de paramètres $t_1, \ldots, t_{N+1}$ trouvés, alors

$$\exists \text{ la distance } \Delta_N \text{ telle que } \forall n = 0 \ldots, N \ : \ d(t_n, t_{n+1}) = \Delta_N \tag{2}$$

pour chacun des nombres de points N il existe une distance Δ_N. Inversement il n'existe pas de nombre N_Δ pour chacun des valeurs Δ qu'on pourrait donner. Mais si on renonce aux égalités strictes dans (2) alors il est possible à partir d'une valeur Δ de chercher un nombre N et des paramètres $t_1, \ldots, t_{N+1}$ afférents qui satisfont approximativement (2). C'est ce nombre qui va être appelé la distance moyenne.

Algorithme 2 : Recherche des $t_1, \ldots, t_N$

Données : les fonctions $x(t), y(t)$, les bornes t_0 et b (qui sera t_{N+1} lorsque N sera connu, la fonction TrouveLeMilieu, la distance moyenne Δ, une précision ϵ
Résultat : N et $t_1, \ldots, t_N$ tels que (2) soit assuré au mieux
début
  In ⟵ [[t_0, b]];
  Out ⟵ [];
  tant que *longueur(In) > 0* **faire**
    [t0,t1] ⟵ In[1];
    In ⟵ reste de In après avoir retranché le premier segment [t0,t1] ;
    distance ⟵ d(t0,t1);
    si *distance* < *facteur* $\times\Delta$ **alors**
      Out ⟵ concaténation de Out avec [t0,t1]
    sinon
      tm ⟵ TrouveLeMilieu(x(t),y(t),t0,t1,ϵ);
      In ⟵ concaténation de In avec [[t0,tm],[tm,t1]]
    fin
  fin
  Out ⟵ second élément des listes de Out;
  Out ⟵ trie de Out;
  Out ⟵ concatenation de t0 avec Out;
  Out en sortie
fin

Question 4 L'algorithme précédent conduit à N points à peu près équirépartis mais pas totalement. écrire un algorithme itératif permettant d'améliorer l'équirépartition sans changer le nombre N de point et qui utilise successivement la formule

$$\text{nouveau–}t_n = \frac{\frac{d(t_{n+1},t_n)}{t_{n+1}-t_n}\, t_{n+1} + \frac{d(t_n,t_{n-1})}{t_n-t_{n-1}}\, t_{n-1}}{\frac{d(t_{n+1},t_n)}{t_{n+1}-t_n} + \frac{d(t_n,t_{n-1})}{t_n-t_{n-1}}} \tag{3}$$

On s'intéressera notamment au critère d'arrêt de cet algorithme itératif.

L'algorithme précédent présente l'inconvénient que la distance entre points successifs n'est pas constante ; il s'agit alors de disposer d'un algorithme gardant le nombre de points N fixe mais améliorant l'uniformité de cette distribution de distance.

Algorithme 3 : Amélioration de l'uniformité de la distance

Données : les fonctions $x(t), y(t)$, une liste de paramètres $t_0, \ldots, t_{N+1}$

Résultat : une nouvelle liste de paramètres $nt_0, \ldots, nt_{N+1}$ améliorant l'uniformité de la distance entre points successifs

début

  pour $n = 1 \ldots N$ **faire**

    $nt_n = \frac{\frac{d(t_{n+1},t_n)}{t_{n+1}-t_n}\, t_{n+1} + \frac{d(t_n,t_{n-1})}{t_n - t_{n-1}}\, t_{n-1}}{\frac{d(t_{n+1},t_n)}{t_{n+1}-t_n} + \frac{d(t_n,t_{n-1})}{t_n - t_{n-1}}}$

  fin

  Sortie de $[t_0, t_1, \ldots, t_N, t_{N+1}]$

fin

L'algorithme 3 est une étape d'un processus itératif dont le critère d'arrêt doit être défini. Pour cela on peut calculer la variance de la distribution de distance comme

$$\frac{1}{N+1}\sum_{n=0}^{N}(d(t_{n+1},t_n))^2 - \left(\frac{1}{N+1}\sum_{n=0}^{N} d(t_{n+1},t_n)\right)^2$$

et décider d'arrêter les itérations dès que l'écart-type correspondant est inférieur à un certain seuil.

Question 5 Comment la formule (3) a-t-elle été trouvée ?

Si le comportement des distances était linéaire comme

$$\begin{aligned} d(t_n, t_{n-1}) &= \alpha\,(t_n - t_{n-1}) \\ d(t_n, t_{n-1}) &= \beta\,(t_{n+1} - t_n) \end{aligned}$$

alors l'application de la formule conduirait à

$$d(\text{nouveau-}t_n, t_{n-1}) = d(\text{nouveau-}t_n, t_{n-1}) = \frac{\alpha\,\beta}{\alpha+\beta}(t_{n+1} - t_{n-1})$$

C'est donc une formule exacte dans le cas linéaire qui est appliqué dans un cas non-linéaire ; c'est ce qu'on appelle une « *linéarisation* . »

Données pratiques Des fonctions de dessin et de recherche de zero par la méthode de dichotomie sont données

```
/* données */
DessineTD1(ListeDesXetY,XYdeT,T,tmin,tmax,xmin,ymin,xmax,ymax):=
 plot2d([[parametric,XYdeT[1],XYdeT[2],[t,tmin,tmax]],[discrete,ListeDesXetY]],
        [x,xmin,xmax],[y,ymin,ymax],[gnuplot_preamble,"set size ratio 1"],
        [style,[lines,1,1],[points,5,2]])$
DTD1:DessineTD1$

ResolutionParDichotomie(Expression,Variable,BorneInf,BorneSup,Precision):=
 block([a:BorneInf,b:BorneSup,e:Precision,x_tilde,m,fm,xx:ev(Variable)],
  block([fa:ev(Expression,ev(Variable)=a),fb:ev(Expression,ev(Variable)=b)],
   if fa=0 then  (x_tilde:a)
   elseif fb=0 then (x_tilde:b)
   elseif fa*fb > 0 then x_tilde:"echec de l'algorithme"
   else
   (while abs(b-a)>e do
    (m:(a+b)/2,fm:ev(Expression,ev(Variable)=m),
    if fm=0 then (a:m,b:m)
```

```
     elseif fa*fm > 0 then (a:m,fa:fm)
     else (b:m,fb:fm)),
     x_tilde:(a+b)/2.0),
    [x_tilde,ev(Expression,ev(Variable)=x_tilde)]))$
RPD:ResolutionParDichotomie$
```

Question 6 Expliquer ce que font

```
DTD1([[0,0],[0.5,0.125],[1,0.5],[%pi,%pi^2/2]],[t,t^2/2],t,0,%pi,0,0,%pi,%pi^2/2);
et
RPD(sin(2*%pi*t)-cos(2*%pi*t),t,0,0.5,1.0e-5);
```

```
/* Question 6 */
DTD1([[0,0],[0.5,0.125],[1,0.5],[%pi,%pi^2/2]],[t,t^2/2],t,0,%pi,0,0,%pi,%pi^2/2);
/* -> dessine d'une part la courbe paramétrée (x(t)=t,y(t)=t^2/2)
entre 0 et pi ; et d'autre part marque les points de coordonnées
((0,0),(0.5,0.125),(1,0.5),(pi,pi^2/2)) ; la zone de tracé est le
rectangle dont le sommet bas gauche a pour cordonnées (0,0) et le
sommet haut droit (pi,pi^2/2) */

RPD(sin(2*%pi*t)-cos(2*%pi*t),t,0,0.5,1.0e-5);
/* -> resoud par dichotomie l'équation sin(2 pi t) = cos(2 pi t) en
partant de l'intervalle [0,0.5] et avec une précision de 10^-5 */
```

Question 7 Écrire en langage de Maxima et sous forme d'une fonction appelée « *TrouveLeMilieu* » l'algorithme de la question 2. Tester TrouveLeMilieu sur

$$\begin{cases} x(t) = \cos t \\ y(t) = \sin t \end{cases} \quad \text{pour a) } [0, \pi/2] \text{ ; b) } [0, 2\pi]$$

```
/* Question 7 */

TrouveLeMilieu(XYdeT,t,t0,t1):=
 block([x0,y0,x1,y1],
 [x0,y0]:ev(XYdeT,ev(t)=t0),
 [x1,y1]:ev(XYdeT,ev(t)=t1),
 ResolutionParDichotomie(
 (XYdeT[1]-x0)^2+(XYdeT[2]-y0)^2-(XYdeT[1]-x1)^2-(XYdeT[2]-y1)^2,t,
 t0,t1,1.0e-5)[1])$
TLM:TrouveLeMilieu$

TLM([cos(t),sin(t)],t,0.0,%pi/2);
/* -> c'est correct */

TLM([cos(t),sin(t)],t,0.0,2*%pi);
/* -> on espérait %pi, mais TrouveLeMilieu ne sait pas le faire */
```

Question 8 Écrire en langage de Maxima et sous forme d'une fonction appelée « *CreeLesTs* » l'algorithme de la question 3. Tester CreeLesTs sur les deux exemples de la question 1 avec le choix d'une distance moyenne de 1.

```
/* Question 8 */
CreeLesTs(XYdeT,t,t0,tN,DistanceMoyenne):=
 block([in:[[float(t0),float(tN)]],out:[],i],
  while length(in) > 0 do
   block([t0:in[1][1],t1:in[1][2],tm],
    in:rest(in),
    block([x0,y0,x1,y1,dst],
     [x0,y0]:ev(XYdeT,ev(t)=t0),
     [x1,y1]:ev(XYdeT,ev(t)=t1),
     dst:sqrt((x1-x0)^2+(y1-y0)^2),
     if dst < sqrt(2.0)*DistanceMoyenne then out:append(out,[[t0,t1]])
     else
      (tm:TrouveLeMilieu(XYdeT,t,t0,t1),
       in:append(in,[[t0,tm],[tm,t1]]))
    )),
  cons(t0,map(lambda([u],u[2]),sort(out))))$
CLT:CreeLesTs$

FaitLesPoints(XYdeT,T,Ts):=
 map(lambda([u],ev(XYdeT,ev(T)=u)),Ts)$
FLP:FaitLesPoints$

xyt:[cos(t),sin(t)];Ts:CLT(xyt,t,0,%pi,1);DessineTD1(FLP(xyt,t,Ts),xyt,t,0,%pi,-1,0,1,2)
/* pas beaucoup de points */
xyt:[cos(t),sin(t)];Ts:CLT(xyt,t,0,%pi,0.1);DessineTD1(FLP(xyt,t,Ts),xyt,t,0,%pi,-1,0,1,
/* un peu plus */

xyt:[t,t^2/2];Ts:CLT(xyt,t,0,%pi,1);DessineTD1(FLP(xyt,t,Ts),xyt,t,0,%pi,0,0,%pi,%pi^2/2
```

Question 9 Écrire en langage de Maxima et sous forme d'une fonction appelée « *AmelioreLaDistribution* » une seule étape de l'algorithme de la question 4. On n'hésitera pas à introduire des fonctions auxiliaires. Tester les améliorations qu'apporte cette étape sur l'exemple très dégradé donné question 6.

non atteint

Question 10 Écrire alors la fonction finale répondant à l'algorithme de la question 4.

non atteint